```c
// 01 FRAME SORT

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct frame {
    int seqno;
    char msg[100];
} m[100];
int main() {
    int n, i, j, r, s[100], temp;
    char ch[100];
    printf("Enter the number of frames:");
    scanf("%d", & n);
    for (i = 0; i < n; i++) {
        s[i] = -1;
        m[i].seqno = -1;
    }
    j = 0;
    while (j < n) {
        r = rand() % n;
        if (s[r] == -1) {
            m[j].seqno = r;
            j = j + 1;
            s[r] = 1;
        }
    }
    for (i = 0; i < n; i++) {
        printf("Enter the message:");
        scanf("%s", m[i].msg);
        srand(i);
    }
    printf("The arrived frames are:\n");
    for (i = 0; i < n; i++) {

        printf("%d\t%s\n", m[i].seqno, m[i].msg);
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - 1 - i; j++) {
            if (m[j].seqno > m[j + 1].seqno) {
                temp = m[j].seqno;
                m[j].seqno = m[j + 1].seqno;
                m[j + 1].seqno = temp;
                strcpy(ch, m[j].msg);
                strcpy(m[j].msg, m[j + 1].msg);
                strcpy(m[j + 1].msg, ch);
            }
        }
    }
    printf("The frames in sorted are :\n Sequence number Message \n");
    for (i = 0; i < n; i++) {
        printf("%d\t%s\n", m[i].seqno, m[i].msg);
    }
    printf("\n");
}



// 02 Distance Vector

#include<stdio.h>
#define INFINITY 999
struct node {
    int cost;
    int via;
```

```
}
c[4][4];
int n;
void findpath(int n1, int n2) {
    int i, t1, t2;
    t1 = c[n1][n2].via;
    if (t1 < n2)
        findpath(n1, t1);
    if (t1 != n2)
        printf("%d-->", t1);
}
void matrix() {
    int i, j, k, cost, x, t;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            cost = INFINITY;
            if (i != j) {
                for (k = 0; k < n; k++) {
                    if (i != k) {
                        x = c[i][k].cost + c[k][j].cost;
                        if (cost > x) {
                            cost = x;
                            t = k;
                        }
                    }
                }
                c[i][j].cost = cost;
                c[i][j].via = t;
            } else {
                c[i][i].cost = 0;
                c[i][i].via = i;
            }
        }
    }
}
main() {
    int i, j, k, x, t, cost = INFINITY;
    int n1, n2, final, t1, t2, next;
    printf("Enter the number of nodes:");
    scanf("%d", & n);
    printf("Enter the edge matrix(enter 999 if nodirect connection)\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("cost[%d][%d]=", i, j);
            scanf("%d", & c[i][j].cost);
            c[i][j].via = INFINITY;
        }
    }
    printf("starting matrix:(Each entry has cost and via node\n");
    for (i = 0; i < n; i++) {
        printf("row %d\t", i);
        for (j = 0; j < n; j++) {
            printf("%d,%d\t", c[i][j].cost, c[i][j].via);
        }
        printf("\n");
    }
    matrix();
    printf("final matrix\n");
    for (i = 0; i < n; i++) {
        printf("row %d\t", i);
        for (j = 0; j < n; j++) {
            printf("%d,%d\t", c[i][j].cost, c[i][j].via);
        }
        printf("\n");
    }
    next = 1;
```

```
    while (next) {
        printf("Enter the two node numbers to find the path\n");
        printf("Enter the source node:");
        scanf("%d", & n1);
        printf("Enter the destination node:");
        scanf("%d", & n2);
        printf("The shortest path to reach %d from %d has cost =%d\n", n2, n1,
c[n1][n2].cost);
        printf("The path is:\n");
        final = c[n1][n2].via;
        printf("%d-->", n1);
        findpath(n1, n2);
        printf("%d", n2);
        printf("would you like to continue (0/1)");
        scanf("%d", & next);
    }
}
```

```
// 03 CRC

#include<stdio.h>
int input[50];
int n;
struct reg {
    int bit;
}
r[16];

int xor(int x, int y) {
    x += y;
    if (x == 0 || x == 2)
        return 0;
}

void compcrc() {
    int lb, x, j, i;
    for (j = 0; j < (n + 16); j++) {
        lb = r[15].bit;
        for (i = 15; i > 0; i--) {
            r[i].bit = r[i - 1].bit;
        }
        r[0].bit = input[j];
        if (lb == 1) {

            r[12].bit = xor(r[12].bit, lb);
            r[5].bit = xor(r[5].bit, lb);
            r[0].bit = xor(r[0].bit, lb);
        }
    }
    printf("Register content:\n");
    for (i = 0; i < 16; i++)
        printf("%d", r[i].bit);
    printf("\n");
    for (x = n, j = 15; j >= 0; x++, j--)
        input[x] = r[j].bit;
    printf("\nThe total message along with crc :\n");
    for (i = 0; i < (n + 16); i++)
        printf("%d", input[i]);
}

int main() {
    int i, j, k, x, y;
    for (i = 0; i < 16; i++)
        r[i].bit = 0;
```

```
    printf("\nEnter the number of bits in the input:\n");
    scanf("%d", & n);
    printf("\nEnter the bits:\n");
    for (k = 0; k < n; k++)
        scanf("%d", & input[k]);
    for (j = n; j < (n + 16); j++)
        input[j] = 0;
    printf("\nAt sender:\n");
    compcrc();
    for (i = 0; i < 16; i++)
        r[i].bit = 0;
    printf("\nThe data is transmitted\n");
    printf("Do you want to introduce error : 0/1 \n");
    scanf("%d", & x);
    printf("=====**************=====\n");
    if (x == 1) {
        for (i = 0; i < n + 16; i++)
            scanf("%d", & input[i]);
    }
    printf("\nAt receiver:\n");
    compcrc();
    if (x == 1) {
        printf("\nThere is an error in the data\n");
        printf("\nThe received data : ");
        for (i = 0; i < n; i++)
            printf("%d", input[i]);
    } else {
        printf("\nThere is no error in the data.\n");
        printf("\nThe received data : ");
        for (i = 0; i < n; i++)
            printf("%d", input[i]);
    }
    printf("\n");
}



// 04 RSA

#include <stdio.h>
typedef unsigned int uint;
uint gcd(uint x, uint y) {
    return y == 0 ? x : gcd(y, x % y);
}
uint multi(uint txt, uint ed, uint n) {
    uint i, rem = 1;
    for (i = 1; i <= ed; i++)
        rem = (rem * txt) % n;
    return rem;
}
short prime(uint no) {
    uint i;
    for (i = 2; i <= no / 2; i++)
        if (no % i == 0) return 1;
    return 0;
}
int main() {
    char msg[100];
    uint pt[100], ct[100], n, d, e, p, q, z, i, len;
    do {
        printf("\nEnter 2 large prime numbers p & q:\n");
        scanf("%d %d", & p, & q);
    } while (prime(p) || prime(q));
    n = p * q;
    z = (p - 1) * (q - 1);
    do {
```

```
        printf("\nEnter prime value of e relative to %d(z):", z);
        scanf("%d", & e);
    } while (gcd(e, z) != 1 || e > n);
    for (d = 2; d < z; d++)
        if ((e * d) % z == 1)
            break;
    printf("Enter the Message\n");
    //get message from keybrd.
    len = read(1, msg, 100) - 1;
    for (i = 0; i < len; i++)
    //store it in plain text array
        pt[i] = msg[i];
    printf("\n Cipher Text=");
    for (i = 0; i < len; i++)
    //convert plain to cipher text
        printf("%d ", ct[i] = multi(pt[i], e, n));
    printf("\n Plain Text=");
    for (i = 0; i < len; i++)
    //convert cipher to plain text
        printf("%c", multi(ct[i], d, n));
}



// 05 LEAKY

#include<stdio.h>
#include<stdlib.h>
    int randomize(int a) {
        int rn = (rand() % 10) % a;
        return rn == 0 ? 1 : rn;
    }

int main() {
    int packet_sz[5], i, clk, b_size, o_rate, p_sz_rm = 0, p_sz, p_time;
    for (i = 0; i < 5; ++i)
        packet_sz[i] = randomize(6) * 10;
    for (i = 0; i < 5; ++i)
        printf("packet[%d]:%d bytes\t", i, packet_sz[i]);
    printf("\nEnter the Output rate:");
    scanf("%d", & o_rate);
    printf("Enter the Bucket Size:");
    scanf("%d", & b_size);
    for (i = 0; i < 5; ++i) {
        if ((packet_sz[i] + p_sz_rm) > b_size)
            if (packet_sz[i] > b_size)
                printf("\n\nIncomming packet size (%d) is Greater than bucket
capacity-PACKET REJECTED", packet_sz[i]);
            else
                printf("\n\nBucket capacity exceeded-REJECTED!!");
        else {
            p_sz_rm += packet_sz[i];
            printf("\n\nIncomming Packet size: %d", packet_sz[i]);
            printf("\nBytes remaining to Transmit: %d", p_sz_rm);
            p_time = randomize(4) * 10;
            printf("\nTime left for transmission: %d units", p_time);
            for (clk = 10; clk <= p_time; clk += 10) {
                sleep(1);
                if (p_sz_rm) {
                    if (p_sz_rm <= o_rate)
                        printf("\n  Packet of size %d Transmitted", p_sz_rm),
                        p_sz_rm = 0;
                    else
                        printf("\n  Packet of size %d Transmitted", o_rate),
                        p_sz_rm -= o_rate;
                    printf("----Bytes Remaining after Transmission: %d", p_sz_rm);
```

```
                } else
                    printf("\n  No packets to transmit!!");
                printf("   Time Left:%d", p_time - clk);
            }
        }
    }
}



// 06 HAMMING CODE


#include<stdio.h>
#include<stdlib.h>
    int power(int x, int y) {
        int i, res = 1;
        for (i = 1; i <= y; i++)
            res = x * res;

        return res;
    }
main() {
    int input[15], m, r = 0, count = 0, i = 0, z = 0, j = 0, n, t[15], k = 0, a,
b, c, cnt = 0, reg = 0;

    for (int i = 0; i < 15; ++i) {
        input[i] = 0;
        t[i] = 0;
    }
    printf("\nenter the number of bits");
    scanf("%d", & m);
    printf("\nenter the %d bits:", m);
    for (i = 0; i < m; i++)
        scanf("%d", & input[i]);
    while (power(2, r) < (m + r + 1))
        r++;
    printf("\nr = %d\n", r);
    for (i = 1; i <= (m + r); i++) {
        if (i == power(2, k)) {
            t[i] = 0;
            k++;
        } else
            t[i] = input[j++];
    }
    printf("\n the actual message is: ");
    for (i = 1; i <= (m + r); i++)
        printf("%d", t[i]);
    n = 1;
    while (n <= power(2, r)) {
        i = n;
        while (i <= m + r) {
            for (j = 0; j < n; j++) {
                if ((i + j) <= (m + r) && t[i + j] == 1) {
                    count++;
                }
            }
            i = i + 2 * n;
        }
        if (count % 2 != 0)
            t[n] = 1;
        n = n * 2;
        count = 0;
    }
    printf("\ndata transmitted       :");
    for (i = 1; i <= (m + r); i++)
```

```
        printf("%d", t[i]);

    printf("\nenter the data transmitted with one bit errror");
    for (i = 1; i <= (m + r); i++)
        scanf("%d", & t[i]);

    printf("\n the errored message is:");
    for (i = 1; i <= (m + r); i++)
        printf("%d", t[i]);
    n = 1;
    while (n <= power(2, r)) {
        i = n;
        while (i <= m + r) {
            for (j = 0; j < n; j++) {
                if ((i + j) <= (m + r) && t[i + j] == 1)
                    cnt++;
            }
            i = i + 2 * n;
        }
        if (cnt % 2 != 0) {
            reg += n;
        }
        n = n * 2;
        cnt = 0;
    }
    if (reg == 0) {
        printf("\nno error");
    } else {
        printf("\nerror in position %d", reg);
        t[reg] = !(t[reg]);
        printf("\n the corrected message is:");
        for (i = 1; i <= (m + r); i++)
            printf("%d", t[i]);
    }
}
```